UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/824,887	04/13/2004	Bhaskar Ghosh	50277-2404	7312

42425 7590 03/04/2009

HICKMAN PALERMO TRUONG & BECKER/ORACLE
2055 GATEWAY PLACE
SUITE 550
SAN JOSE, CA 95110-1083

EXAMINER	
HWA, SHYUE JIUNN	
ART UNIT	**PAPER NUMBER**
2163	
MAIL DATE	**DELIVERY MODE**
03/04/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/824,887
Filing Date: April 13, 2004
Appellant(s): GHOSH ET AL.

Christopher M. Tanner
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 11/18/2008 appealing from the Office action mailed 06/19/2008.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The Examiner is not aware of any related appeals, interferences, or judicial proceedings, which directly affect or be affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

There are no un-entered amendments.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

US Patent No. 6,289,334 B1	Reiner	01/31/1997
US Patent No. 5,495,606 A	Borden et al.	11/04/1993
US Patent Application No. 2004/0268227 A1	Brid	06/27/2003
US Patent No. 5,857,180 A	Hallmark et al.	07/21/1997

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 101

3. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefore, subject to the conditions and requirements of the title.

Claims 15-25 and 27-29 are rejected under 35 U.S.C.101 because the claims fail to place the invention squarely within one statutory class of invention. On page 25-26, paragraph 0079-0081 of the instant specification, applicant has provided evidence that applicant intends the “medium” to include Punchcard, SIGNAL and transmission Media. As such, the claim is drawn to a form of energy. Energy is not one of the four categories of invention and therefore this claim(s) is/are not statutory. Energy is not a series of steps or acts and thus is not a process. Energy is not a physical article or object and as such is not a machine or manufacture. Energy is not a combination of substances and therefor not a composition of matter.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1-7, 11, 14-21, 25 and 29 are rejected under 35 U.S.C. 103(a) as being unpatentable over Reiner et al. (US Patent No. 6,289,334 B1, hereinafter “Reiner) in view of Borden et al. (US Patent No. 5,495,606 A, hereinafter “Borden”).

As to claims 1 and 15, Reiner teaches the claimed limitations:

“A method for processing a database statement within a database server” as a method for digital data processing paralleling the operation of the digital data processing system (column 4, lines 60-65). The specific method used for merging or ordering the subquery results is completely dependent on the nature of the query. The existence of aggregate functions, ORDER BY, or GROUP BY clauses (e.g. database statement) are the main factors to consider (column 23, lines 26-35).

“receiving at the database server the database statement” as a database management system adapted to access data records stored in said database store, said database management system including a standard interface adapted to receive a query and to apply that query to said stored data records to generate a result (claim 1).

“determining that at least one operation required by the database statement can be parallelized” as Parallel ORACLE Program Interface (PUPI) routines would determine whether a particular call required parallel processing within the database server (column 31, lines 24-39).

“generating a set of information about how to execute the database statement” as the information from the optimizer including possible tables to be chosen as the driving table, can be obtained from data files generated by the DBMS in connection with the query, and accessed by use of the EXPLAIN command (column 9, lines 34-44).

A decomposable query it generates a set of subqueries, each of which is based on the initial query but which is directed to data in one or more respective of the partitions of database. Then element initiates and invokes threads which initiate execution of the subqueries (column 7, line 64 to column 8, line 51).

It may also be possible to automate the choice of partitioning table. This avoids having to depend on the application programmer to correctly determine which queries can be effectively parallelized and how to do it (column 13, lines 40-54).

"Causing a plurality of slave processes to perform said at least one operation by sharing the set of information with each slave process of said plurality of slave processes" as the subcursor pnode functionality could potentially be decomposed to more than one specialized pnode types, but need not be. It is unique among pnode types described thus far in having two executor functions which share the same pnode data structure (column 42, lines 16-25).

"Wherein the set of information shared with each slave process includes
(a) information about a task to be performed by said slave process, and
(b) information about one or more tasks, to be performed by processes other than the slave process, to execute the database statement" as bind descriptor for the root cursor. This describes any host parameters referenced in the original input query which has been decomposed. It is modified each time the pcursor is re-opened. Since host variables described in the bind descriptor are not modified by query execution, and since they are referenced identically in all parallel subqueries of the same pcursor the

root cursor's bind descriptor can be shared by parallel subqueries (column 60, lines 31-61).

Although Reiner teaches psubqries place their output values in different locations, which may change from fetch to fetch, their output columns otherwise share the same description. We can economize on memory by separating out the sharable portions of the descriptor information, which could be collected in the vanilla descriptor (column 63, lines 1-16).

Reiner does not explicitly teach the claimed limitation “sending to each slave process of said plurality of slave processes data that indicates which part of the set of information shared with the slave process represents the part of the at least one operation that should be performed by the slave process”.

Borden teaches the splitter receives the query and, if possible, splits it into multiple queries. The resulting queries are then passed to the scheduler which schedules them for processing on an appropriate slave processor so as to minimize the total time spent processing the original query (column 7, lines 50-58; see also figure 3).

The shared access to the entire database from any processor in the query processor complex provides another performance boost because all the processors and all of the database management systems can process any query against the database; there are no processing or queueing delays such as may be experienced in a system where the database is partitioned such that only certain processors can access certain parts of the database (column 2, lines 52-64).

The queries are received by the master query processor and passed to the splitter function that determines whether the database query can be efficiently split. If so, the database query is split up and scheduled for processing so as to achieve maximum performance improvement; if not, the database query is submitted as-is to one of the CPCs in the query processor complex via the high-speed channel connections among them. Server tasks in each CPC take the database queries (either split or unsplit) and send them to the local database manager for execution. The resulting answers are sent back to the query processor complex master processor (column 5, lines 19-39; see also figure 4).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made, having the teachings of Reiner and Borden before him/her, to modify Reiner shared the slave process because that would allow use of multiprocessors for more processing capacity and parallel execution capabilities as taught by Borden (col. 2, lines 7-10).

As to claims 2 and 16, Reiner teaches the claimed limitations:

"the step of sharing the set of information includes sharing an execution plan for the database statement; and sharing the execution plan with a particular slave process of the plurality of slave processes is performed by" as the subcursor pnode functionality could potentially be decomposed to more than one specialized pnode types, but need not be. It is unique among pnode types described thus far in having two executor functions which share the same pnode data structure. The master executor is called by

the subcursor pnode's parent. The primary job of the master executor is to spawn a parallel thread to run the parallel executor, when the subcursor pnode is first pulled in an UNINITIALIZED state. The parallel executor in turn starts an ORACLE session and opens an ORACLE cursor for the parallelized subcursor (column 42, lines 16-46).

“providing an original statement of the database statement to a node on which the particular slave process resides, wherein the original statement is the form of the database statement in which the database statement was received by the database server; at said node, generating an equivalent execution plan based on the original statement; and the particular slave process accessing the equivalent execution plan” as query decomposition is done by making a number of copies of the original query, and then appending additional predicates to each subquery to make it match one of the existing partitions of one of the tables in the query. These subqueries are then executed in parallel. Finally, a combining query (or function) over the subquery results produces the result of the original query (column 12, lines 38-46).

Also, Borden teaches all the central processing complexes (CPCs) in the query processor complex are running a database manager that is operating in read-only mode (e.g. original statement). In this embodiment, the database manager is DB2 operating in a shared read-only data configuration; however, those skilled in the art will understand that the database manager could be any hierarchical or relational database manager operating in read-only mode. This means that all of these processors can access shared database information in read-mode only (column 5, lines 54-67).

As to claims 3 and 17, Reiner teaches the claimed limitations:

"providing to the node additional information that includes at least one of (a) values associated with session parameters of a database session in which the database statement was received, and (b) values associated with optimizer parameters that were used by an optimizer to generate a plan for the database statement in a node other than said node; and the step of generating an equivalent execution plan is performed based, at least in part, on the additional information" as bind descriptor for the root cursor. This describes any host parameters referenced in the original input query which has been decomposed. It is modified each time the pcursor is re-opened. (ORACLE permits re-opening a cursor to bind new host parameter values, without an intervening close. This causes the same user-visible behavior as if there were an intervening close, but the query does not have to be re-parsed and re-optimized.) Since host variables described in the bind descriptor are not modified by query execution, and since they are referenced identically in all parallel subqueries of the same pcursor (unless we choose to specify fileid through a host parameter), the root cursor's bind descriptor can be shared by parallel subqueries (column 60, lines 30-60).

As to claims 4 and 18, Reiner teaches the claimed limitations:

"generating a set of information includes generating an execution plan for the database statement, wherein the set of information includes the execution plan; and the step of sending to each slave process of said plurality of slave processes data that indicates which part of the at least one operation should be performed by the slave

process includes sending to each slave process data that indicates a specific portion of the execution plan that is to be performed by the slave process" as queries joining two or more tables, the decomposer generates corresponding subqueries by duplicating the query and appending a predicate for matching records in the corresponding table partition of the driving table, which is selected by the decomposer based on the access strategy chosen by the query optimizer portion of the DBMS. Those skilled in the art will appreciate that information from the optimizer including possible tables to be chosen as the driving table, can be obtained from data files generated by the DBMS in connection with the query, and accessed by use of the EXPLAIN command (column 9, lines 33-44). (e. g. The EXPLAIN command produces a complete description of a query's execution plan. The plan describes all possible alternative plans a query can choose from at execution time. These plans show you which indexes and join operations a query might use).

As to claims 5 and 19, Reiner teaches the claimed limitations:

"sending to each slave process data that indicates a specific portion of the execution plan that is to be performed by the slave process includes sending to a particular slave process data that indicates a particular portion of the execution plan that is to be performed by the particular process; and the method further includes the step of the particular slave process determining how to execute the particular portion based, at least in part, on characteristics of the execution plan other than the particular portion of the plan that is to be executed by the particular slave process" as the operation of

assembler on results generated by the UPI of DBMS and threads in response to the subquery signals. More particularly, the drawing shows that for intercepted queries that call for aggregate data functions, element performs a like or related data function of the results of the subqueries. Thus, for example, if the intercepted query seeks a minimum data value from the database table--and, likewise, the subqueries seek the same minimum value from their respective partitions--then element generates a final result signal representing the minimum among those reported to the assembler by the DBMS and threads. For queries that combine aggregate and non-aggregate functions, a combination of elements are invoked. For queries involving grouping operations, the decomposer generates corresponding subqueries by duplicating the query, along with the grouping clause in its predicate list (column 9, line 45 to column 10, line 30; see also figure 4).

As to claims 6 and 20, Reiner teaches the claimed limitations:
“generating a set of information includes generating an execution plan for the database statement; constructing a shared cursor for the database statement, wherein the shared cursor provides access to the execution plan; and the step of sharing access includes providing each slave process of said plurality of slave processes access to the shared cursor” as the subcursor pnode functionality could potentially be decomposed to more than one specialized pnode types, but need not be. It is unique among pnode types described thus far in having two executor functions which share the same pnode data structure. The master executor is called by the subcursor pnode's parent. The primary

job of the master executor is to spawn a parallel thread to run the parallel executor, when the subcursor pnode is first pulled in an UNINITIALIZED state. The parallel executor in turn starts an ORACLE session and opens an ORACLE cursor for the parallelized subcursor (column 42, lines 15-46).

As to claims 7 and 21, Reiner teaches the claimed limitations:
"providing each slave process of said plurality of slave processes access to the shared cursor includes allowing two or more of said slave processes to access a shared instance of the shared cursor" as while not strictly necessary, it would be useful to represent any common expression by a single EXPR subtree, and share that subtree by pointing to it from each place it is referenced. For example, two expressions in FIG. 24 with a single instance of the EXPR for PRICE pointed to by both the > and < operators. Doing this when generating the parse tree can save us a lot of trouble each time we need to determine if two expressions reference the same subexpression, while we are using the tree (column 52, lines 26-45) and (column 42, lines 15-46).

As to claims 11 and 25, Reiner teaches the claimed limitations:

"generating a set of information includes generating an execution plan for the database statement, wherein the set of information includes the execution plan; and the method further comprises the step of inserting into the execution plan a granule iterator row source that encapsulates a horizontal partitioning of a base object upon which the database statement operates" as it may be desirable to implement some cases entirely

by means of combining functions, and others entirely by means of combining queries. However, it is preferrable to combine the two approaches by encapsulating combining query behavior inside pnodes. In a combining queries approach, the output rows from parallelized subcursors are inserted into one or more temporary intermediate tables. A combining query is formed, which can be handed to ORACLE to execute against the intermediate table(s), producing an output stream which mimics that which the original query would have produced if handed direct to ORACLE (column 47, lines 15-60).

If rows with matching key values could be clustered together, then using an index would reduce the total I/O in a much wider variety of cases, again, with or without QD. This is essentially what ORACLE clusters accomplish. Now, if instead of clustering rows with a given key value into one clump, they could be clustered in N clumps (column 28, lines 45-55).

As to claims 14 and 29, Reiner teaches the claimed limitations:

"generating a set of information includes generating an execution plan for the database statement, wherein the set of information includes the execution plan; and the method further comprises the step of inserting into the execution plan a parallelizer row source that encapsulates the scheduling of tasks that slave processes are to perform" as in a combining queries approach, the output rows from parallelized subcursors are inserted into one or more temporary intermediate. A combining query is formed, which can be handed to ORACLE to execute against the intermediate table(s), producing an

output stream which mimics that which the original query would have produced if handed direct to ORACLE (column 47, lines 16-34).

Reiner does not explicitly teach the claimed limitation “encapsulates the scheduling of tasks that slave processes are to perform”.

Borden teaches the query splitter and scheduler provide an additional performance boost by parallelizing the individual queries so that the resulting multiple split queries can be processed Concurrently on the multiple CPCs (column 2, lines 31-35; see also element 33 of figure 3).

6. Claims 8-10 and 22-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Reiner et al. (US Patent No. 6,289,334 B1) as applied to claims 1 and 15 above, and further in view of Borden et al. (US Patent No. 5,495,606 A) and Brid (US Patent Application No. 2004/0268227 A1, hereinafter “Brid”).

As to claims 8-10 and 22-24, Although Reiner teaches share subtree by pointing to it from each place it is referenced (column 52, lines 26-67). The subcursor pnode functionality could potentially be decomposed to more than one specialized pnode types, but need not be. It is unique among pnode types described thus far in having two executor functions which share the same pnode data structure (column 42, lines 16-46).

Reiner does not explicitly teach the claimed limitation “providing each slave process of said plurality of slave processes access to the shared cursor includes allowing one of the slave processes to access a first instance of the shared cursor, and allowing another one of the slave processes to access a second instance of the shared

cursor", "one slave process resides on a first node; the other slave process resides on a second node; and the first node is a different node than said second node" and "a first plurality of slave processes on said first node share access to said first instance of said shared cursor; and a second plurality of slave processes on said second node share access to said second instance of said shared cursor".

Brid teaches because of the concerns of memory consumption for a collection of a large amount of Row Objects, example embodiments allow for sharing Table Row Object's 205 characteristics among a plurality of rows within the Collection of Rows 220. A Row Object 205 can be shared if the characteristics for each individual Cell Object 215 within a Collection of Cells 230 of the Row Object 205 can be deduced from the owning Row Object 105 and owning Column Object 210. For example, if Cell Object 215 has a state that is selected, then either Column Object 210 and/or Row Object 205 must also be selected in order for the characteristics of Row Object 205 to be shared among several rows. By contrast, if Column Object 210 is unselected and Row Object 205 is also unselected, but Cell 215 is selected then the characteristics of Row Object 205 cannot be shared and memory must be allocated for representing the instance of Row Object 205 (page 3, paragraph 0029; see also figure 2).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made, having the teachings of Reiner, Borden and Brid before him/her, to modify Reiner providing each slave process of said plurality of slave processes access to the shared cursor because that would allow reducing memory requirements as taught by Brid (page 1, paragraph 0010).

7. Claims 12-13 and 27-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over Reiner et al. (US Patent No. 6,289,334 B1) as applied to claims 1 and 15 above, and further in view of Borden et al. (US Patent No. 5,495,606 A) and Hallmark et al. (US Patent No. 5,857,180 A, hereinafter "Hallmark").

As to claims 12-13 and 27-28, Reiner teaches the claimed limitations:

"generating a set of information includes generating an execution plan for the database statement, wherein the set of information includes the execution plan; and the method further comprises the step of inserting into the execution plan at least one distribution row source that specifies how data is to be redistributed between one of a first slave set and a query coordinator; and a first slave set and a second slave set" as by combining the DBMS's indexing and hashing mechanisms in the manner described above, the aforementioned scatter clustering technique achieves a good mix of I/O parallelism and hit ratio. It does this by storing the data records using the DBMS's hash-based storage techniques with abnormally small bucket size, thereby distributing small bucket-size clusters of related information around the disk, and by retrieving the data using the DBMS's indexing mechanism (column 11, lines 17-26).

"inserting into the execution plan at least one distribution row source includes: inserting into the execution plan at least one sender-side distribution row source that indicates how sending processes are to distribute data that the sending processes produce; and inserting into the execution plan at least one receiver-side distribution row source that indicates how receiving processes are to obtain data that the receiving

processes are to consume" as when pulled to return a row, it would pull rows from its children and insert them in the appropriate intermediate table until all children returned EOD, and would then open its combining cursor over the intermediate table(s), and fetch and return rows from that cursor (column 47, line 61 to column 48, line 10).

Reiner does not explicitly teach the claimed limitation "distribution row source between one of a first slave set and a query coordinator".

Hallmark teaches that use table queues to partition and transport rows between sets of processes. A table queue (TQ) encapsulates the data flow and partitioning functions (column 3, lines 30-40).

A TQ provides data flow directions. A TQ can connect a Query Coordinator (QC) to a Query Server (QS). For example; a QC may perform a table scan on a small table and transmit the result to a table queue that distributes the resulting rows to one or more QS threads. The table queue has one input thread and some number of output threads equaling the number of QS threads (column 10, lines 47-60).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made, having the teachings of Reiner, Borden and Hallmark before him/her, to modify Reiner inserting into the execution plan at least one distribution row source because that would allows blocks of data to be efficiently transmitted between systems as taught by Hallmark (column 7, lines 48-52).

(10) Response to Argument

Appellant Argues:

(1) Regarding transmission media, it is true that the specification gives examples of transmission media. However, this is irrelevant because the claims have been amended to cover only "storage" media, thereby excluding transmission media. It is improper to reject a claim under 35 U.S.C. § 101 for subject matter that the claim doesn't even cover in claims 15-25 and 27-29.

Examiner Responds:

Examiner respectfully disagrees. In response to applicant's argument, although the claims amended to cover only "storage" media, the instant specification disclose a modem local to computer system can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector can receive the data carried in the infra-red signal and appropriate circuitry can place the data on bus. Bus carries the data to main memory, from which processor retrieves and executes the instructions. The instructions received by main memory may optionally be stored on storage device either before or after execution by processor (page 26, paragraph 81). The received code may be executed by processor as it is received, and/or stored in storage device, or other non-volatile storage for later execution. In this manner, computer system may obtain application code in the form of a carrier wave (page 28, paragraph 85).

Consequently, the rejection to claims 15-25 and 27-29 under 35 U.S.C.101 is maintained.

Appellant Argues:

(2) Neither Reiner nor Borden has any equivalent for the claimed "sharing the set of information with each slave process of said plurality of slave processes".

Examiner Responds:

Examiner respectfully disagrees. In response to applicant's argument, Borden teaches in a read-only, shared-data (e.g. information) configuration, the configuring the database to be read-only and providing access to the entire database from any processor and any database management system in the query processor complex provides additional performance benefits (column 2, lines 39-43).

The master query processor in the embodiment contains the query-splitter while the query splitter needs to be able to communicate with the front-end system. The scheduler schedules the split queries for processing in parallel on the various CPCs (central processing complexes) of the query processor complex and sends the split queries to the server tasks on the designated slave query processors (column 4, line 57 to column 5, line 7; see also elements 31-33 of figure 3). All the direct access storage device database storage accessible to the query processor complex is accessible from any CPC in the complex via the high-speed channel connections through the switches. That is, all the database storage is shared by all the systems in the query processor complex (column 5, lines 40-53; see also elements 24A-24H of figure 2).

A query is sent to the master processor via the high speed channel from the front-end processor. The splitter receives the query and, if possible, splits it into multiple

queries. The resulting queries are then passed to the scheduler which schedules them for processing on an appropriate slave processor so as to minimize the total time spent processing the original query (column 7, lines 50-61).

Also, Reiner teaches the subcursor pnode functionality could potentially be decomposed to more than one specialized pnode types, but need not be. It is unique among pnode types described thus far in having two executor functions which share the same pnode data structure. The master executor is called by the subcursor pnode's parent (column 42, lines 16-46).

Appellant Argues:

(3) Reiner's host variables being "referenced identically in all parallel subqueries" is a good example of what Applicant is not claiming. Note specifically Claim 1 section (b), and the language "information about one or more tasks, to be performed by processes other than the slave process, to execute the database statement". This claimed feature is contradicted by Reiner's language "referenced identically".

Examiner Responds:

Examiner respectfully disagrees. In response to applicant's argument, Reiner teaches rather than being routed directly to DBMS, the query is intercepted by the parallel user program interface (PUPI or parallel interface). Element 74A (responsible for decomposing the query) routes queries not susceptible to decomposition to DBMS, but for a decomposable query it generates a set of subqueries, each of which is based on the initial query but which is directed to data in one or more respective of the

partitions 72A, 72B, 72C of database 72. Then element 74A initiates and invokes threads 78A, 78B, 78C, which initiate execution of the subqueries. The subqueries corresponding to threads 78A, 78B, 78C are routed to the user program interface (UPI or standard interface) of DBMS 76. Multiple subqueries are preferably applied to the UPI of DBMS 76 in parallel with one another, thus capitalizing on the database partitions and on the multiprocessing nature of the preferred digital data processing system. Each thread routes its subquery to a separate server process in DBMS (column 7, line 64 to column 8, line 51; see also figure 3B, non-decomposed queries on the top of the drawing, e.g. other than the slave process 78A, 78B, 78C).

Also, Borden teaches the shared access to the entire database from any processor in the query processor complex provides another performance boost because all the processors and all of the database management systems can process any query against the database; there are no processing or queueing delays such as may be experienced in a system where the database is partitioned such that only certain processors can access certain parts of the database (column 2, lines 52-63).

The splitter receives the query and, if possible, splits it into multiple queries. The resulting queries are then passed to the scheduler which schedules them for processing on an appropriate slave processor (20B-20H) so as to minimize the total time spent processing the original query (column 7, lines 54-61; see also figure 3). A slave processor server receives the split query and routes it to its local database manager for processing. Database managers are well known. This embodiment uses the DB2 database manager in all instances (column 9, lines 8-20; see also figure 5).

splitting said complex read-only (e.g. shared) into two or more query elements and scheduling each of said two or more query elements for execution by one of the one or more slave CPCs, each one of said slave CPCs independently executing a database management system, each one of said CPCs comprising each of said one or more slave CPCs being capable of accessing any part of a database using a slave channel means (claim 1). one or more of said slave CPCs, each one of said slave CPCs controlled by an operating system and each one of said slave CPCs comprising first slave channel means for communicating with said master CPC, and second slave channel means for accessing any portion of said database (claim 5).

Appellant Argues:

(4) This is significant because within Borden, information/queries are always split, but are never "shared". Borden's slave processors 20B-20H and 52-53 never share any information, and thus cannot suggest the claimed step in claim 1.

Examiner Responds:

Examiner respectfully disagrees. In response to applicant's argument, Borden teaches all the CPCs in the query processor complex are running a database manager that is operating in read-only mode. In this embodiment, the database manager is IBM DATABASE 2 (DB2) operating in a shared read-only data (e.g. information) configuration, a mode of operation provided in DB2 Version. However, those skilled in the art will understand that the database manager (DBM) could be any hierarchical or relational database manager operating in read-only mode. This means that all of these

processors can access shared database information in read-mode only. One of the front-end systems runs a database manager program that owns the database information and can update it (column 5, lines 54-67).

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the Examiner in the Related Appeals and Interferences section of this Examiner's Answer.

(12) Conclusion

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

James Hwa

Conferees:

/James Hwa/
Examiner, Art Unit 2163

/C. T. T./

Primary Examiner, Art Unit 2169

/don wong/

Supervisory Patent Examiner, Art Unit 2163

/Charles Rones/

Supervisory Patent Examiner, Art Unit 2164